Overlapping Community Search for Social Networks

Arnau Padrol-Sureda [*1], Guillem Perarnau-Llobet [†2], Julian Pfeifle [*3], Victor Muntés-Mulero [†4]

[*]*Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya*
C/Jordi Girona, 1-3, 08034 Barcelona, Catalonia
[1]arnau.padrol@upc.edu
[3]julian.pfeifle@upc.edu

[†]*DAMA-UPC, Universitat Politècnica de Catalunya*
C/Jordi Girona, 1-3, 08034 Barcelona, Catalonia
[2]perarnau@ac.upc.edu
[4]vmuntes@ac.upc.edu

***Abstract*— Finding decompositions of a graph into a family of clusters is crucial to understanding its underlying structure. While most existing approaches focus on partitioning the nodes, real-world datasets suggest the presence of overlapping communities. We present OCA, a novel algorithm to detect overlapped communities in large data graphs. It outperforms previous proposals in terms of execution time, and efficiently handles large graphs containing more than 10^8 nodes and edges.**

I. Introduction and Prior Work

As the importance and use of social networks increases at an ever faster pace, the need to understand and analyze their structure becomes more and more pressing. Given their modelization as huge graphs that can contain billions of entities, studying the structure of these graphs is key to understanding the properties of the networks. In particular, many applications need to comprehend the underlying community structure of the graph to infer its topology and the relation between its elements, cf. [3], [4], [6], [14].

The presence of nodes belonging to several communities arises naturally from real data [5], [12]: a person probably belongs to the communities representing his group of friends, job partners, family, etc. However, most of the proposals from the graph clustering literature do not admit overlapping communities [2], [6], [7], [11], [13], [14], [15].

The first attempts to unveil the overlapping community structure of a graph appear in [8], [12]. The algorithm CFinder is presented in [12]. It is based on retrieving all cliques of the graph; however, this operation turns out to be prohibitive for large graphs. A more efficient algorithm is [8], which finds communities by maximizing a certain fitness function. We refer to it using the initials of the authors, LFK.

In this short paper we present a novel algorithm to find the overlapping communities in graphs, OCA. It is based on the optimization of a new fitness function for evaluating the quality of a community. The theoretical background used to deduce the function can be found in Section II, and the function itself is described in Section III. Section IV outlines our algorithm OCA. Section V contains the results of executing OCA on several different large graphs containing up to 10^8 nodes and edges (graph extracted from the Wikipedia), as well as a comparison in terms of quality and execution time to the most relevant algorithms presented in the literature. Finally, Section VI sketches some future directions.

II. The Vectorial Search Space

The search space of communities considered by our algorithm consists of all subsets of nodes of a simple undirected graph $G = (V, E)$ on n nodes. We envisage all these subgraphs as elements in a high-dimensional vector space $\mathbb{R}^d$, because this helps us to intuitively deduce a natural function φ to find communities. More precisely, we map each node v of G to a vector $v \in \mathbb{R}^d$ (using the same letter to denote both), and a subset of nodes to the sum of its corresponding vectors. We must emphasize that we *never* explicitly construct neither these vectors nor their sums: the prohibitive amount of memory this would consume would prevent us from managing large graphs.

Lovász [10] in 1979 suggested the following way to represent graphs in a vector space.

Definition 1: A collection $\mathcal{V} = \{v_1, \ldots, v_n\}$ of unit vectors in some real vector space is a *virtual vector representation* of G if there exists $0 \leq c < 1$, such that $\langle v_i, v_j \rangle = c$ whenever $\{i, j\} \in E$, and $\langle v_i, v_j \rangle = 0$ whenever $\{i, j\} \notin E$.

Definition 2: The *search space* Γ associated to $\mathcal{V}$ is the graph with one vertex for each nonempty subset $S \subseteq V$ (identified with the corresponding sum vector $\sum_{i \in S} v_i$), and edges between subsets that differ in only one element.

Because Γ is so large ($2^n - 1$ nodes), it would not be reasonable to generate it explicitly. Instead, we use it as a tool to formalize the process of maximizing a fitness function to find communities.

Example 1: Figure 1 shows the virtual vector representation of a graph. Since y and z are connected, but x and t are not, the angle $\angle(y, z)$ between y and z is smaller than $\angle(x, t) = \frac{\pi}{2}$, so $y + z$ is longer (further away from 0) than $x + t$. ◇

This example suggests the *squared Euclidean length*,

$$\varphi(S) \;=\; \left\| \sum_{i \in S} v_i \right\|^2,$$

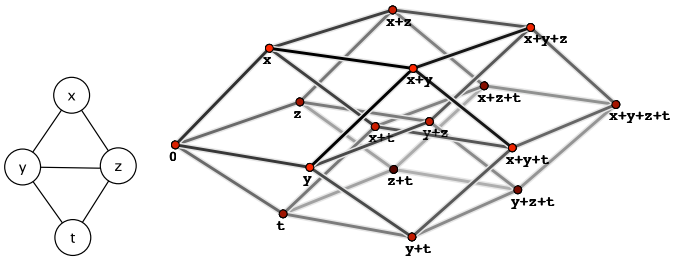

Fig. 1. A graph and its virtual vector representation, together with the search space Γ formed by all sum vectors.

as a first candidate for a fitness function for the maximization. The following example reinforces this idea.

Example 2: If $\mathcal{V}$ is a virtual vector representation of G and S an independent subset of size k (meaning that two nodes in S are never connected by any edge), then $\varphi(S) = k$. On the other hand, the sum vector corresponding to a complete subgraph K_k can be seen to have squared length

$$ck^2 + (1-c)k \;=\; \Theta(k^2).$$

This different behavior becomes more striking as c grows. $\diamond$

This leads us to expect that the best-connected subsets will be maxima of this function. But that is not true, because φ *always* grows when the subset increases. So there exists only one maximum, the entire graph G (just as Figure 1 suggests).

On the other hand, Example 2 shows us that larger values of c make it easier to distinguish communities. Therefore, we are interested in the largest possible value of c compatible with G. It can be shown that this largest admissible value is $c = -1/\lambda_{\min}$, with $\lambda_{\min}$ the most negative eigenvalue of the adjacency matrix A of G. This value can be efficiently calculated using the well-known power method.

III. The directed Laplacian

We turn Γ into $\Gamma^{\uparrow}$ by orienting each edge in Γ toward the node representing the largest subset. For example, in Figure 1 all edges point from left to right.

Our definitive fitness function is based on the variation of the previous function φ over the graph Γ. To measure this, we introduce a notion of derivative in a graph through a new operator, the *directed Laplacian*.

This operator is based on the classical Laplacian from graph theory, see for example [1]. While the classical Laplacian is strongly related to a second order derivative, our new approach uses first order derivatives to model the increment of φ.

Definition 3: The value at v of the *directed Laplacian* of a function f on the directed graph $\Gamma^{\uparrow}$ is

$$\mathcal{L}_{\Gamma^{\uparrow},f}(v) \;=\; f(v) - \sum_{u:u\to v} \frac{f(u)}{\sqrt{\mathrm{indeg}(v)\;\mathrm{indeg}(u)}}.$$

It evaluates the differences between the value of f on a node and its incoming neighbors. Setting $f = \|\cdot\|^2$ and $s = |S|$, the directed Laplacian $\mathcal{L} := \mathcal{L}_{\Gamma^{\uparrow},\varphi}(S)$ evaluates to

$$\mathcal{L} \;=\; s - \sqrt{s(s-1)} + 2c\,\mathrm{E}_{\mathrm{in}}(S)\left(1 - \frac{s-2}{\sqrt{s(s-1)}}\right),$$

where $\mathrm{E}_{\mathrm{in}}(S)$ counts the edges in G with both ends in S.

IV. The Algorithm

The *Overlapping Community Search*, OCA, is the algorithm we propose to retrieve the communities of the graph, i.e. the local optima of the fitness function.

OCA has been devised to find each community independently, so it repeatedly uses the same procedure to obtain different communities from different randomly distributed initial seeds.

In particular, it starts with a random neighborhood of the seed. Then it greedily adds (removes) the node whose addition (removal) to the set implies the greatest increment of the fitness function $\mathcal{L}$. When we cannot add nor remove any node without worsening its fitness, we have found a local maximum, and thus a community.

This procedure is then repeated until a halting criterion is met. The fact that we accept community structures where not all nodes belong to a community (so just the most relevant nodes are included in a community) makes it important to design a non-trivial halting criterion. However, the discussion of the halting criterion is outside the scope of this paper, as well as the selection of the initial set.

An issue we have encountered when applying OCA is the frequent apparition of communities that are "too similar", i.e. that differ in very few nodes. To avoid this situation we postprocess the results by merging these communities.

Apart from that, in some cases we may need to include all nodes into at least one community. In these situations, we just assign each "orphan node" to the community to which most of its neighbors belong.

V. Results

In this section, we present the results of several experiments that we have run to test two different aspects of our proposal: quality of results and execution time. Moreover, we compare our results with the most efficient proposals for overlapping community search [8], [12]. Table I shows the tested datasets.

TABLE I
DATASETS ANALYZED BY OCA

Name	# nodes	# edges
LFR-benchmark	10^4–10^6	$\sim 10^5$–10^7
Daisy	10^5	$\sim 4 \cdot 10^5$
Wikipedia	16 986 429	176 454 501

- *LFR-benchmark*: The authors of [9] present a new benchmark for community detection without overlapping. We use it since we know which communities are to be found, something that is not clear in real networks. The generation of these graphs depends on various parameters which we set to default values. For example, the mixing parameter μ determines how many edges each node shares with nodes from other communities, and thus the overall sharpness of the community structure.

- *Daisy trees*: We propose these overlapped graphs because, to our knowledge, there exists no benchmark allowing overlapping in the literature. These graphs are composed of several *daisy flowers* (called in this way due to their shape, Figure 4), joined by their petals to form a tree. A *daisy* defined by parameters $p, q, n \in \mathbb{N}$ and $\alpha, \beta \in [0,1]$ has vertices indexed by $\{0, \ldots, n-1\}$, distributed into $p-1$ petals and a core. The vertices of the i-th petal, for $1 \leq i \leq p-1$, are those whose indices are congruent to $i \bmod p$, while the set of vertices in the core is $\{v : v = 0 \bmod p\} \cup \{v : v = 0 \bmod q\}$. Notice that each vertex whose index v satisfies both $v \neq 0 \bmod p$ and $v = 0 \bmod q$ lies in both a petal and the core. To create the daisy graph, add each edge in the petals or the core with probability α, respectively β. Finally, a *daisy tree* with parameters $k \in \mathbb{N}$ and $\gamma \in [0,1]$ is grown from an initial daisy by executing the following procedure k times: Generate a new daisy, and attach it to a random daisy already in the tree by randomly selecting two petals, and adding edges between them with probability γ.
- Wikipedia: We test our algorithm on a real dataset. Wikipedia articles (nodes) have links (edges) to their translation or other related articles.

The experiments are performed on a single processor at 2.83 GHz. The operating system is Linux (kernel version 2.6.26). Graphs are managed with C++ structures created ad hoc for this problem. Of all the experiments we ran, the most memory-consuming ones were those using the Wikipedia graph, requiring around 2.5 Gb of RAM. Therefore, all the experiments presented can be run on a regular personal computer.

A. Quality Analysis

To test the quality of the communities found by OCA, we use the generated graphs whose communities we know beforehand: the LFR-benchmarks and Daisy Trees. To test the difference between the real and found community structures, we define the *similarity* ρ between communities C and D as

$$\rho(C,D) \;=\; 1 - \frac{|C \setminus D| + |D \setminus C|}{|C \cup D|}. \qquad \text{(V.1)}$$

Using ρ, we can compare two community structures $F = \{F_1, \ldots, F_\ell\}$ and $O = \{O_1, \ldots, O_m\}$ as follows. Denote by $V_i = \{O_j \mid \operatorname{argmax}_k \rho(F_k, O_j) = i\}$ the subset of communities of O that fit better with F_i than with the other F_k. The *suitability* of the observed community structure O with respect to the real structure F is

$$\Theta(F,O) \;=\; \frac{1}{\ell} \sum_{i=1}^{\ell} \frac{1}{|V_i|} \sum_{O_j \in V_i} \rho(F_i, O_j). \qquad \text{(V.2)}$$

The function Θ takes values between 0 and 1, where 1 implies exactly the same community structure, and 0 indicates a totally different one. Note that this function is defined even for overlapping structures.

In the following example we analyze the behavior of Θ against the mixing parameter μ with different algorithms to search for overlapping communities. The LFR-benchmarks set the value of μ between 0 and 0.5 if a community structure is wanted. Above 0.5, there are no clear communities, and $\mu \geq 1$ yields a completely random graph.

We test the algorithms OCA, LFK and CFinder [8], [12]. We used the standard parameter $\alpha = 1$ in LFK. In CFinder the value of the parameter k that yielded the best results is $k = 3$, and it is the one we used. As our postprocessing techniques, presented in Section IV, also improve the quality of the other algorithms, we applied them to all the results.

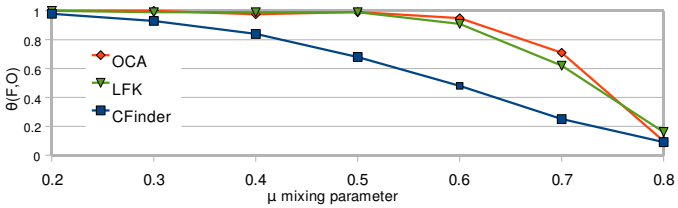

Fig. 2. Evolution of Θ against μ.

Figure 2 shows that OCA finds almost exactly the community structure for $\mu \leq 0.5$, and is reliable for $\mu \leq 0.7$. With LFK we obtain similar measures, while with CFinder the community retrieval does not reach the same level of accuracy.

As the previous benchmarks do not produce overlapping communities, we will now test the algorithms against the Daisy Tree benchmarks with different graph sizes. In Figure 3 we can see the values of $\Theta(D,O)$, cf. (V.2), where D is the daisy community structure. We can observe that both LFK and CFinder perform worse than OCA for overlapping communities.

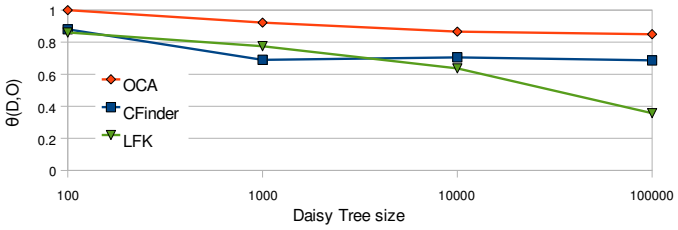

Fig. 3. Θ of daisy community structure with different sizes

To understand these results, in Figure 4 we show the common communities each of the algorithms finds (we say common because their creation involves certain randomness, so the communities are not always the same).

B. Execution Time Analysis

A second aspect we want to study is the performance of the different algorithms. First, we test the scalability of the algorithms against LFR-datasets created with the same parameters and different sizes.

Figure 5 shows a comparison of the running times of the three algorithms using the LFR-benchmark. For the algorithms we use the standard parameters, and we do not run any post-processing. We have chosen the parameters specified in the caption in order to generate graphs with cliques of a

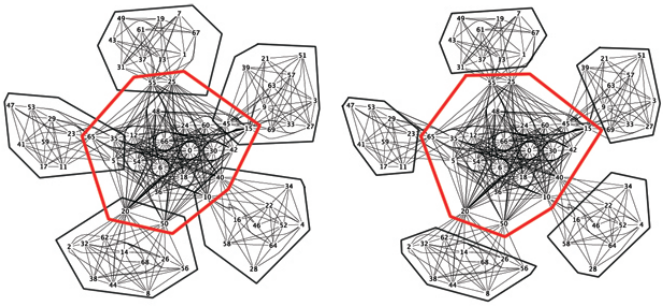

Fig. 4. Typical communities found in the daisy graph. *Left:* OCA and CFinder, *Right:* LFK

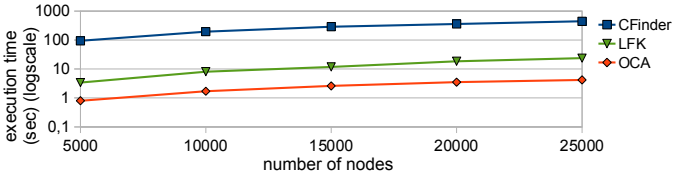

Fig. 5. Execution time of a graph generated with LFR-benchmark with av.deg.=50, max.deg.=150, min.com.size=500 and max.com.size=700

moderate size, since CFinder needs to find cliques to start its community search, and does not support giant cliques. Nevertheless, CFinder is prohibitively slow even taking into account these observations, so we discard it for experiments on larger graphs.

Another advantage of OCA as compared to other proposals is its support of big communities. To prove this claim, we performed the following experiment. We created a series of LFR-benchmark graphs whose communities had sizes in the intervals $[k, k+50]$ for different values of k. Figure 6 shows the results for OCA and LFK. CFinder was not able to perform these experiments in a reasonable time.

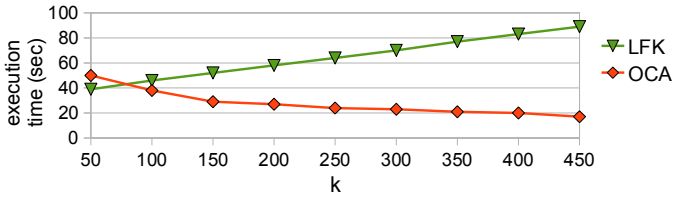

Fig. 6. Execution times of OCA and LFK on graphs generated with LFR-benchmark with av.deg.=50, max.deg.=150, min.com.size=k and max.com.size=$k + 50$

Finally, we ran OCA on the Wikipedia dataset, and found all relevant communities in less than 3.25 hours.

VI. Conclusions and Future Work

We have presented a novel way of finding communities that it is not only efficient for handling large graphs, but also takes into account the fact that nodes in a graph might belong to several communities at the same time. For this, we have introduced a mapping from graphs to vector spaces, which is a pioneering technique for community search. Our results show that OCA is better able to find the overlapping community structure of graphs. We also show that OCA substantially improves the execution time of existing proposals, making it suitable for large networks.

Future work will involve performing the complexity analysis on OCA to rigorously predict the time each execution will take. Moreover, now that the communities are identified, we will explore the hierarchies and relations among them. Finally, this work enables us to pioneer neighboring areas, such as graph summarization for graphs containing overlapped communities.

Acknowledgments

We want to express our gratitude to the authors of [8], [12] who have kindly provided the implementations of their respective algorithms.

The authors from DAMA-UPC want to thank Generalitat de Catalunya for grant number 2009 SGR-01187 and Ministerio de Educacion y Ciencia of Spain for grant TIN2009-14560-C03-03. Arnau Padrol thanks the PhD support grant 2009FI_A 00050 of the Generalitat de Catalunya and the European Social Fund. Julian Pfeifle acknowledges support from MEC grants MTM2008-03020 and MTM2009-07242 and Gen. Cat. DGR 2009SGR104.

References

[1] F. Chung. *Spectral Graph Theory*. American Mathematical Society, February 1997.
[2] L. Donetti and M. A. Munoz. Detecting network communities: a new systematic and efficient algorithm, October 2004.
[3] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72(2):027104+, 2005.
[4] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35:66–71, 2002.
[5] S. Fortunato and C. Castellano. Community structure in graphs, 2007.
[6] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99:7821, 2002.
[7] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515, 2004.
[8] A. Lancichinetti, S. Fortunato, and J. Kertesz. Detecting the overlapping and hierarchical community structure of complex networks, 2008.
[9] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 78(4), 2008.
[10] L. Lóvasz. On the Shannon capacity of a graph. *IEEE Trans. Inform. Theory*, 25:1–7, 1979.
[11] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69:066133, 2004.
[12] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814, 2005.
[13] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proc. Natl. Aacd. Sci. USA*, 101:2658, 2004.
[14] M. Sales-Pardo, R. Guimer, A. Moreira, and L. Amaral. Extracting the hierarchical organization of complex systems. *Proc. Natl. Acad. Sci. USA*, 104:15224–15229, 2007.
[15] S. van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, May 2000.